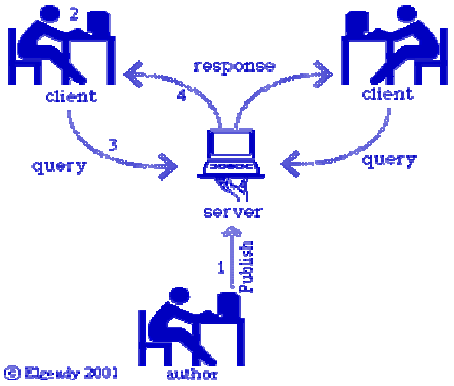
Chương 2 : Tầng Ứng Dụng

Mục tiêu:

- Định nghĩa, Hoạt động các giao thức trong tầng Ứng dụng.
  - Mô hình Khách hàng - Người phục vụ (client-server)
  - Mô hình đồng đẳng
  - Tìm hiểu một số giao thức tầng Ứng dụng thông qua các ví dụ cụ thể.



Sẽ học cái gì ?

- Các giao thức cụ thể:
  - HTTP
  - FTP
  - SMTP
  - POP
  - DNS
- Lập trình các ứng dụng mạng : socket API

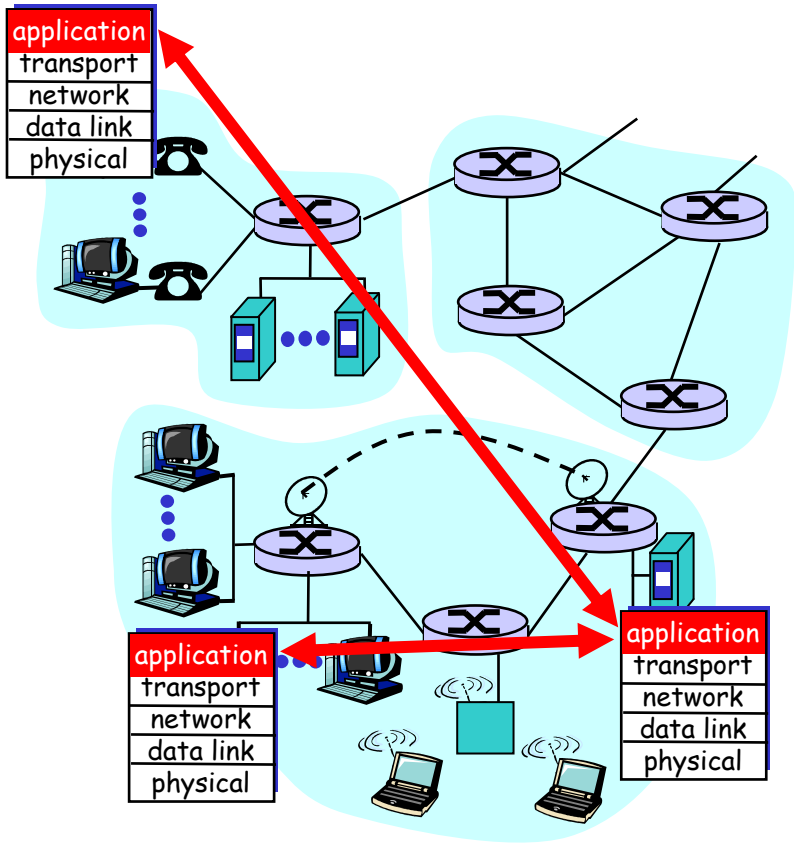
Trình Ứng dụng và Giao thức tầng Ứng dụng

Trình ứng dụng : Các tiến trình phân tán truyền thông với nhau

- Chạy trên các thiết bị đầu cuối
- Trao đổi thông điệp với nhau.
- Email, FTP, Web

Giao thức tầng ứng dụng

- Là một phần của trình ứng dụng.
- Xác định thông điệp trao đổi giữa các ứng dụng.
- Sử dụng dịch vụ truyền thông do giao thức tầng dưới (TCP, UDP) cung cấp.



Một số thuật ngữ

- **Tiến trình :** là chương trình chạy trên thiết bị đầu cuối.
- Trên cùng máy tính, hai tiến trình có thể truyền thông với nhau thông qua **truyền thông liên tiến trình** (HĐH quản lý).
- Các tiến trình chạy trên các máy tính khác nhau: tuân thủ giao thức tầng ứng dụng.

- User agent: phần mềm đóng vai trò giao diện giữa người dùng và mạng.
  - Cài đặt Giao thức tầng Ứng dụng.
  - Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

Mô hình Client-Server

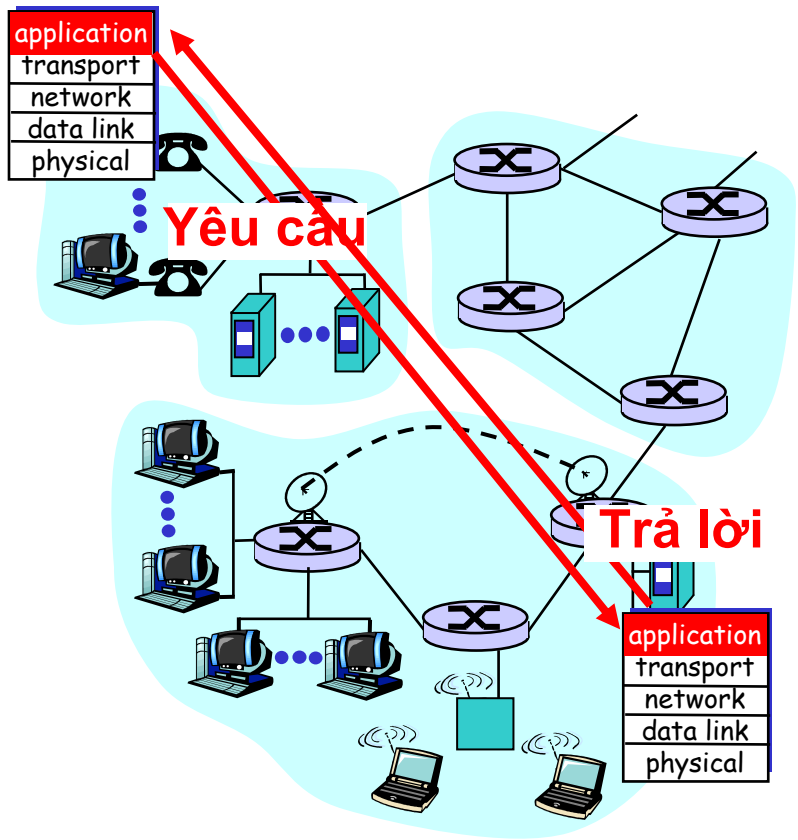
Ứng dụng gồm hai thành phần :
Client và *Server*

Client:

- Khởi tạo giao tiếp với server ("speaks first")
- Yêu cầu dịch vụ từ server.
- Web: client nhận và hiển thị web page; e-mail: các trình đọc thư.

Server:

- Cung cấp dịch vụ theo yêu cầu từ Client
- Web server gửi Web page, Mail server gửi và nhận E-mail



Giao thức tầng Ứng dụng

API: Application Programming Interface

- Là giao diện giữa tầng Ứng dụng và tầng Giao vận.
- Socket : Internet API
  - Hai tiến trình truyền thông bằng cách gửi/nhận dữ liệu vào/từ socket.

Vậy thì Phân biệt các tiến trình bằng cách nào?

- **Địa chỉ IP** của máy.
- **Số hiệu cổng** – cho phép bên máy tính nhận xác định tiến trình nào nhận thư.

Còn tiếp …

Ứng dụng cần Dịch vụ Giao vận gì?

Mất mát dữ liệu (Data loss)

- Một số ứng dụng có thể chấp nhận một số mất mát.
- Một số ứng dụng lại đòi hỏi 100% dữ liệu truyền tin cậy (web).

Thời gian (Timing)

- Một số ứng dụng (hội thoại qua Internet, các trò chơi trực tuyến) đòi hỏi độ trễ thấp.

Băng thông (Bandwidth)

- Một số ứng dụng (đa phương tiện) yêu cầu băng thông tối thiểu để có thể hoạt động được.
- Một số ứng dụng khác có thể sử dụng bất cứ băng thông nào được cấp phát.

Yêu cầu với một số Ứng dụng cụ thể

Ứng dụng	Mất mát	Băng thông	Thời gian
Truyền file	Không	Co dãn	Không
Thư tín điện tử	Không	Co dãn	Không
Web	Có thể	Co dãn	Không
Đa phương tiện Thời gian thực	Có thể	audio: 5Kb-1Mb video:10Kb-5Mb	Có, 100's msec
stored audio/video	Chấp nhận	same as above	Có, few secs
Trò chơi tương tác	Chấp nhận	few Kbps up	Có, 100's msec
Ứng dụng Tài chính	Không	Co dãn	Không xác định

Dịch vụ Giao thức Giao vận Internet

Dịch vụ TCP:

- ***Hướng nối*** *:* Yêu cầu thiết lập kết nối giữa client và server.
- ***Truyền dữ liệu tin cậy*** giữa tiến trình gửi và nhận
- ***Điều khiển lưu lượng*** *:* bên gửi sẽ không làm "lụt" bên nhận.
- ***Kiểm soát tắc nghẽn****:* điều chỉnh tốc độ gửi khi mạng quá tải.
- **Không hỗ trợ** *:* thời gian, băng thông tối thiểu.

Dịch vụ UDP:

- Truyền dữ liệu không tin cậy giữa các tiến trình gửi và nhận.
- **Không hỗ trợ:** thiết lập kết nối, độ tin cậy, điều khiển lưu lượng, kiểm soát tắc nghẽn, thời gian, băng thông tối thiểu.

Tại sao một số ứng dụng sử dụng UDP

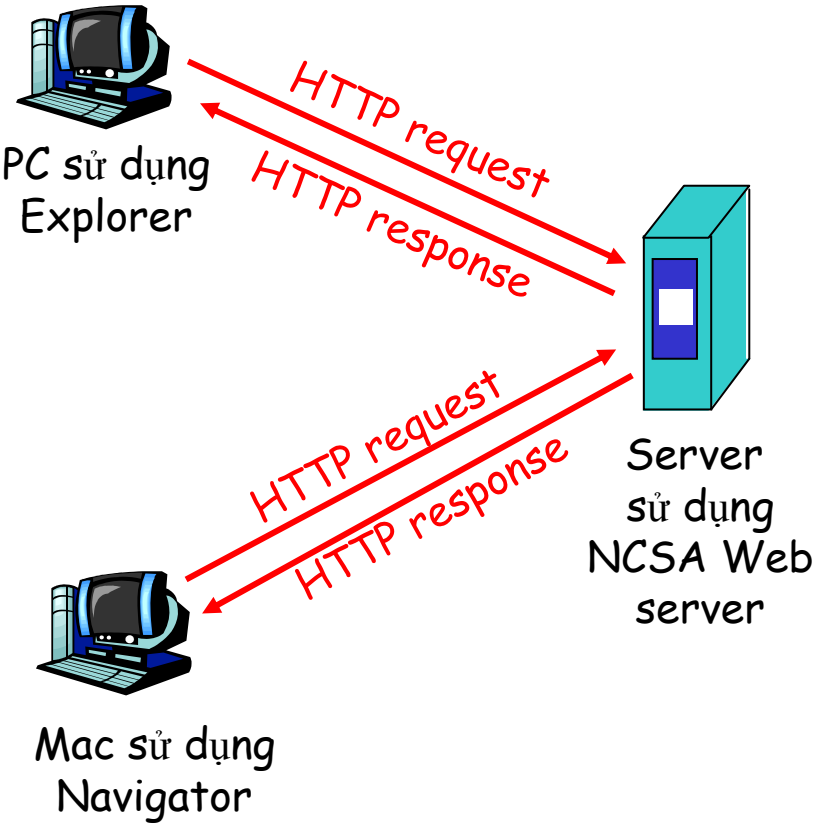
Các ứng dụng trên Internet và giao thức giao vận tương ứng

Ứng dụng	Giao thức Ứng dụng	Giao thức Giao vận tương ứng
Thư tín Điện tử	SMTP [RFC 821]	TCP
Truy cập từ xa	telnet [RFC 854]	TCP
Web	HTTP [RFC 2068]	TCP
Truyền file	FTP [RFC 959]	TCP
Đa phương tiện luồng	Độc quyền (vd. RealNetworks)	TCP hoặc UDP
File server ở xa	NFS	TCP hoặc UDP
Điện thoại Internet	Độc quyền (vd. Vocaltec)	Thường là UDP

Giao thức Web : HTTP
(Hyper Text Transfer Protocol)

HTTP: Giao thức truyền siêu văn bản.

- Là giao thức tầng ứng dụng cho ứng dụng Web.
- Sử dụng mô hình client/server
  - ***client:*** browser yêu cầu, nhận, hiển thị các đối tượng Web.
  - ***server:*** Web server gửi các đối tượng khi có yêu cầu
- HTTP1.0: RFC 1945
- HTTP1.1: RFC 2068



Giao thức HTTP

HTTP sử dụng giao thức TCP.

- Client khởi tạo kết nối TCP (socket) tới server qua cổng 80
- Server chấp nhận kết nối TCP từ Client
- Các thông điệp HTTP : trao đổi giữa Browser (HTTP client) và Web server (HTTP server)
- Đóng kết nối TCP.

HTTP là giao thức "không trạng thái"

- server không lưu lại thông tin về yêu cầu của client

Giao thức lưu lại trạng rất phức tạp !

- Các trạng thái trước đây phải được lưu lại (tốn bộ nhớ).
- Nếu kết nối server/client bị gián đoạn, trạng thái trên chúng có thể sai khác => cần cơ chế điều chỉnh lại

Ví dụ về HTTP

Bạn đánh địa chỉ trên trình duyệt
www.someSchool.edu/someDepartment/home.htm

(bao gồm file HTML tham chiếu tới 10 ảnh JPEG)

1a. HTTP client khởi tạo kết nối TCP tới HTTP server tại địa chỉ **www.someSchool.edu**. Cổng mặc định là **80**.

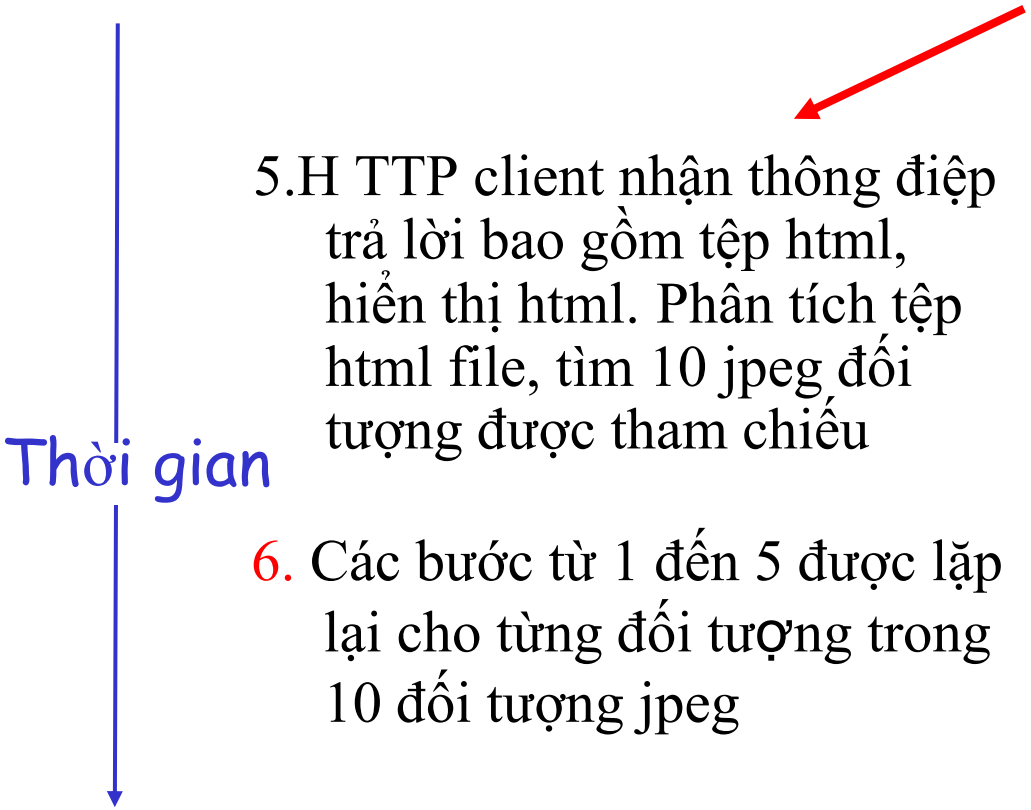
1b. HTTP server ở địa chỉ www.someSchool.edu đợi kết nối TCP ở cổng **80**, chấp nhận kết nối, thông báo lại cho client.

2. HTTP client gửi thông điệp HTTP yêu cầu (bao gồm URL) qua kết nối TCP vừa thiết lập

3. HTTP server nhận thông điệp yêu cầu,lấy các đối tượng được yêu cầu gửi vào trong thông điệp trả lời, (**someDepartment/home.htm**) gửi thông điệp vào socket

Thời gian

Ví dụ về HTTP



4. HTTP server đóng kết nối TCP.

5.H TTP client nhận thông điệp trả lời bao gồm tệp html, hiển thị html. Phân tích tệp html file, tìm 10 jpeg đối tượng được tham chiếu

6. Các bước từ 1 đến 5 được lặp lại cho từng đối tượng trong 10 đối tượng jpeg

Kết nối liên tục và không liên tục

Không kiên trì:

- HTTP/1.0: server phân tích yêu cầu, trả lời rồi đóng kết nối TCP.
- Một đối tượng: 2 RTT (độ trễ).
- Mỗi lần truyền, chịu một độ trễ do thiết lập kết nối.
- ***Nhưng phần lớn trình duyệt mở đồng thời nhiều kết nối TCP.***

Kiên trì:

- Mặc định cho HTTP/1.1
- server phân tích yêu cầu, trả lời, phân tích yêu cầu kế tiếp: trên cùng một kết nối TCP
- client gửi yêu cầu cho tất cả các đối tượng khi nhận được file HTML cơ sở
- Ít bước hơn, nhanh hơn.

Định dạng Thông điệp Yêu cầu HTTP

- Có hai kiểu thông điệp HTTP : **Thông điệp yêu cầu** (request) và **Thông điệp trả lời** (response)
- Thông điệp HTTP request:
  - Theo định dạng mã ASCII.

Dòng yêu cầu (Lệnh GET, POST, HEAD)

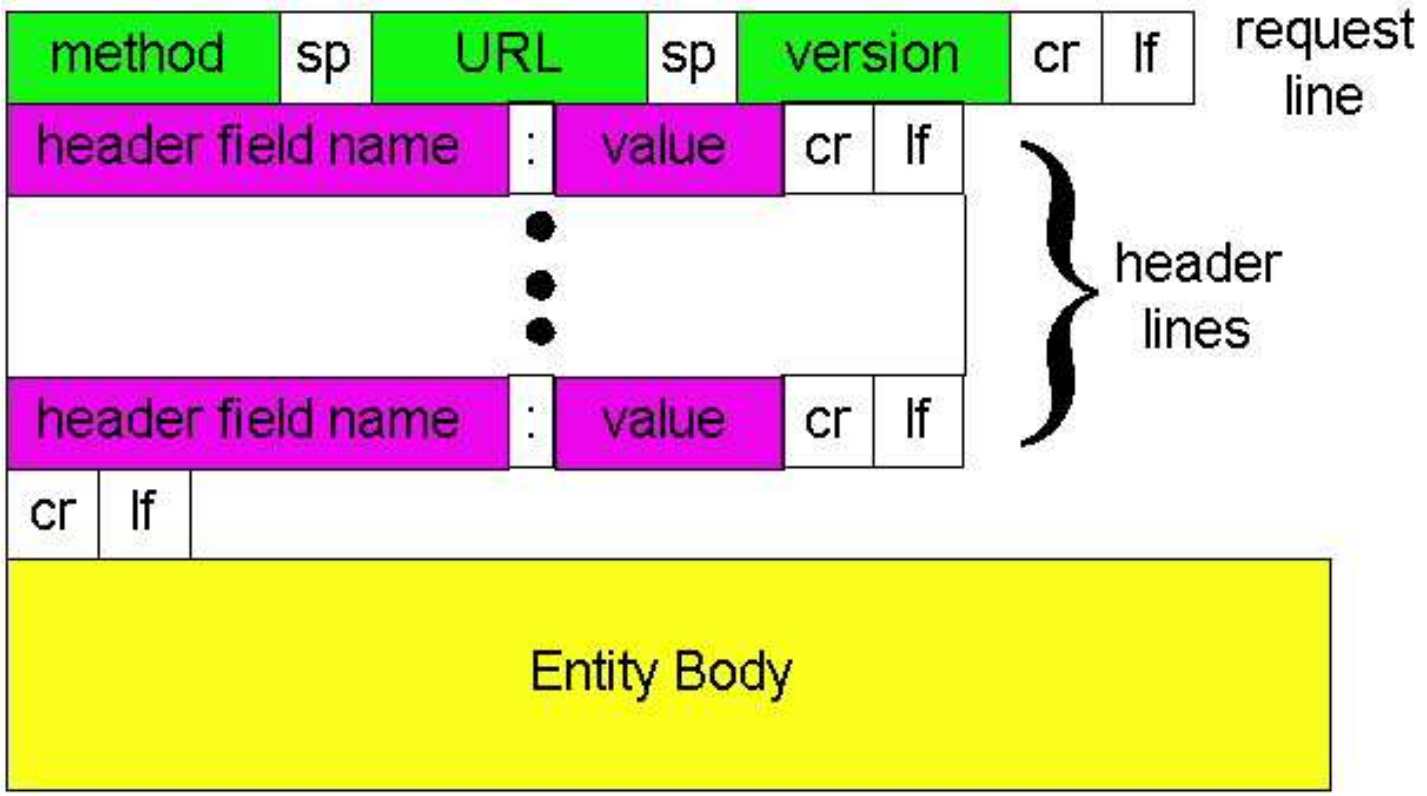
Các dòng header

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

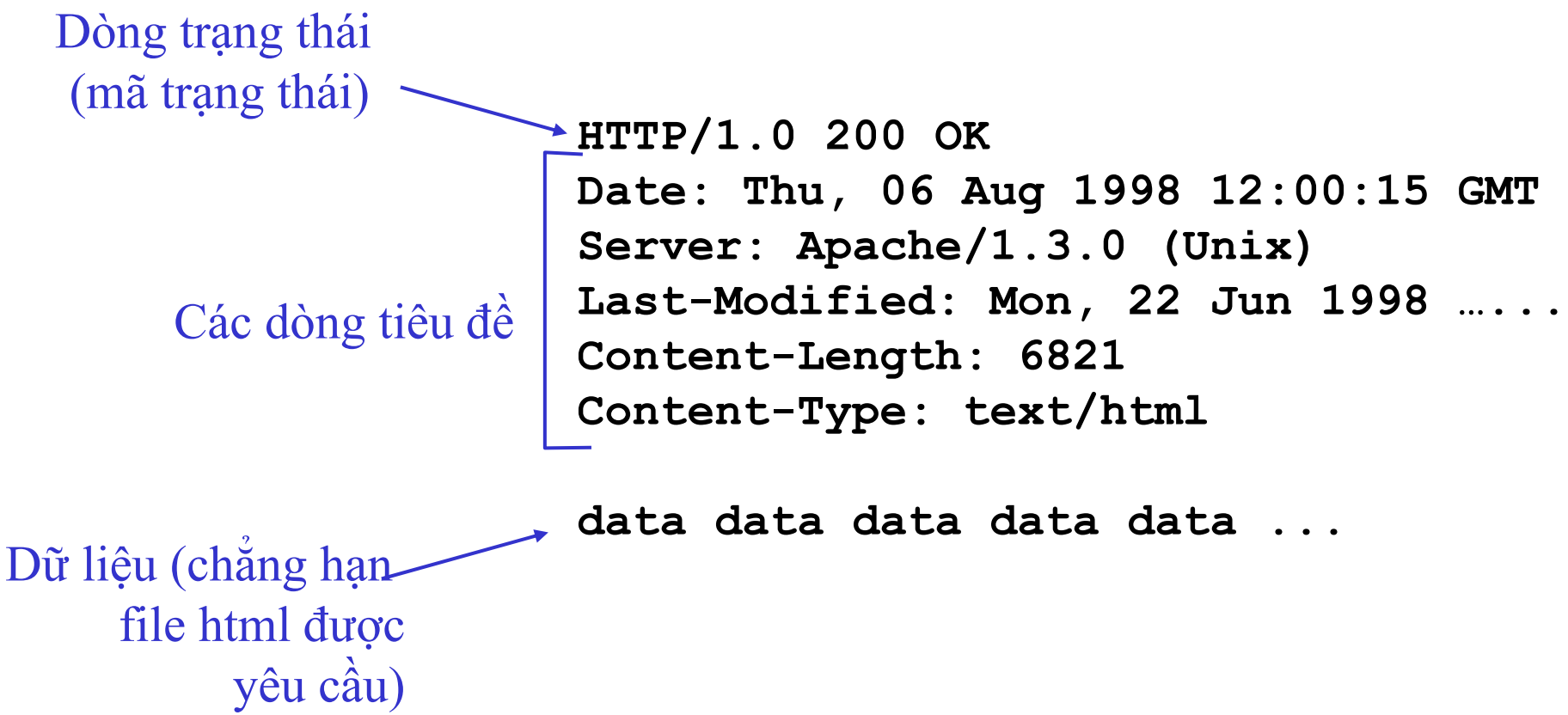
CR,LF: Kết thúc thông điệp

(CR,LF)

Định dạng Thông điệp Yêu cầu HTTP



Định dạng Thông điệp Trả lời HTTP



Mã trạng thái trong Thông điệp Trả lời

Được ghi ở dòng đầu tiên trong thông điệp Server trả lời Client

Một số mã thường gặp:

200 OK

- Yêu cầu thành công, các đối tượng được yêu cầu ở phần thân thông điệp.

301 Moved Permanently

- Đối tượng yêu cầu đã được chuyển và địa chỉ mới của đối tượng được đặt trong trường Location:

400 Bad Request

- Server không hiểu được Thông điệp yêu cầu

404 Not Found

- Đối tượng được yêu cầu không có trong server

505 HTTP Version Not Supported

- Server không hỗ trợ phiên bản giao thức HTTP.

Thử kiểm nghiệm HTTP (phía Client)

1. Telnet tới Web server:

```
telnet www.eurecom.fr 80
```

Tạo kết nối TCP tới cổng 80
(cổng mặc định cho HTTP server)
ở địa chỉ www.eurecom.fr

2. Lệnh GET trong thông điệp HTTP request:

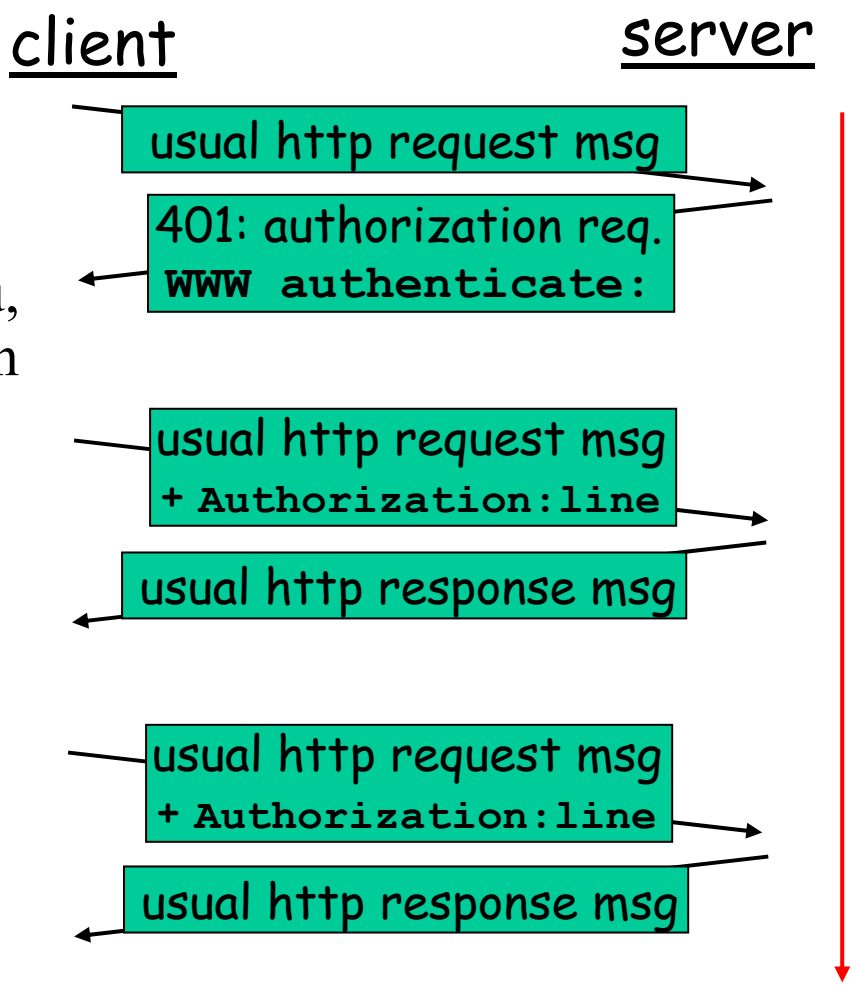
```
GET /~ross/index.html HTTP/1.0
```

Gửi thông điệp yêu cầu lấy tệp
index.html trong thư mục ~ross
cho client

3. **Xem thông điệp response do HTTP server gửi về!**

Tương tác User-Server: Kiểm chứng

- Mục tiêu : kiểm soát quyền truy cập tới đối tượng lưu trên server.
- Phương pháp : sử dụng Tên truy cập và Mật khẩu
- Không trạng thái : mỗi lần yêu cầu, client phải chứng tỏ mình có quyền
- Kiểm chứng :
  - Tiêu đề **`authorization:`** trong mỗi yêu cầu gửi đi.
  - Nếu không có quyền : server từ chối truy cập và gửi yêu cầu có trường tiêu đề **`WWW authenticate:`**



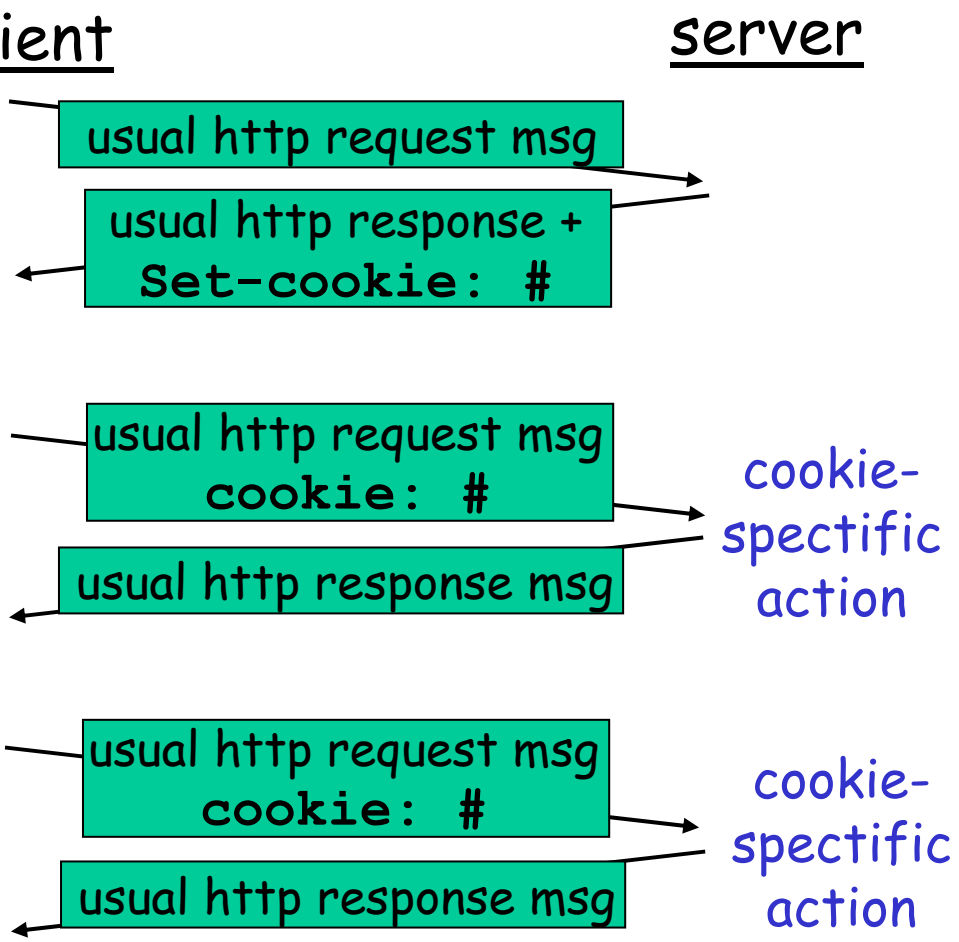
Thông thường, trình duyệt cache username/pass để người dùng không phải gõ lại trong mỗi truy cập

Lưu lại trạng thái nhờ Cookie

- Do Server tạo ra, lưu lại để sử dụng về sau:
  - Thẩm định quyền
  - Ghi nhớ các "sở thích", hoạt động của người dùng.
- Server gửi "cookie" tới client trong thông điệp response.

  Set-cookie: 1678453
- Client gửi kèm cookie trong những lần yêu cầu sau.

  cookie: 1678453

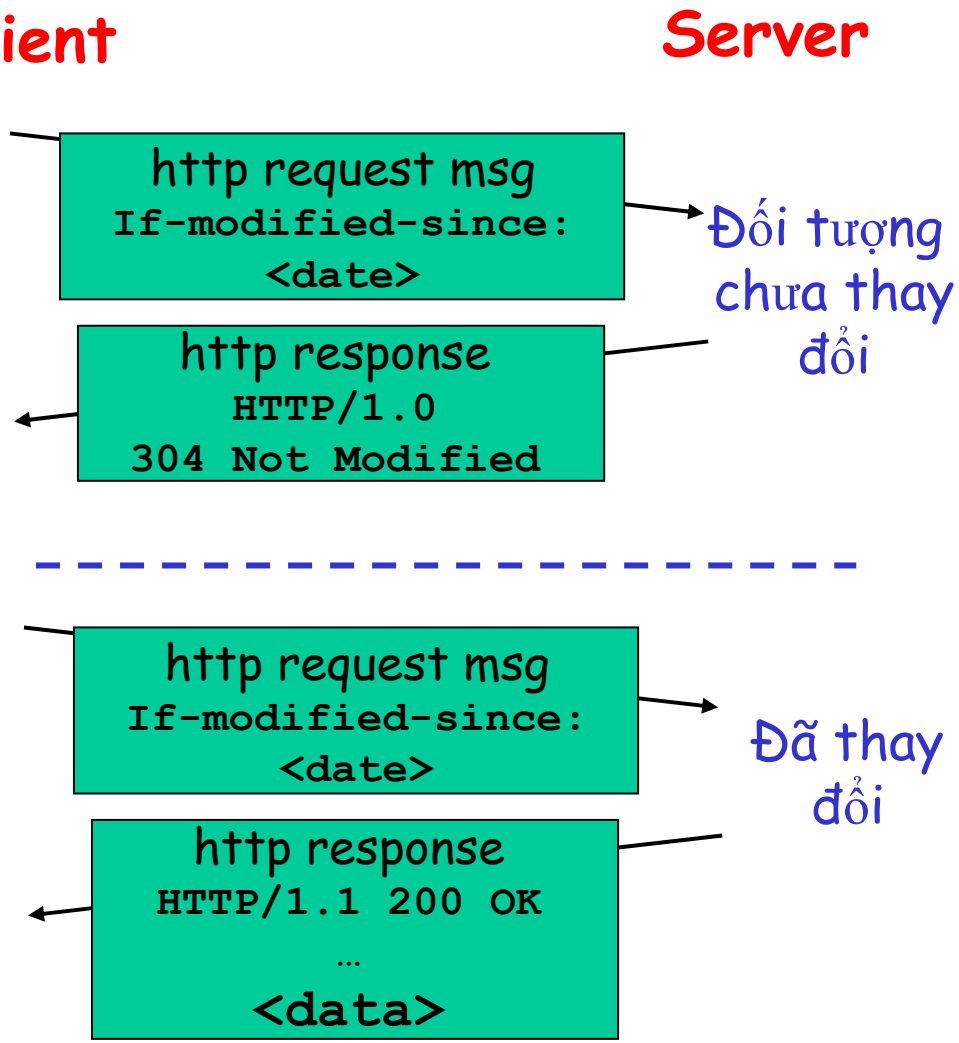


GET có điều kiện: caching bên client

- **Mục đích** : không gửi lại nếu client đã có bản mới nhất của đối tượng trong cache.
- Client: chỉ ra ngày lấy đối tượng về trong thông điệp HTTP request.

  If-modified-since: <date>
- Server: Thông điệp trả về sẽ không kèm theo đối tượng nếu đối tượng đó chưa bị thay đổi trên server:

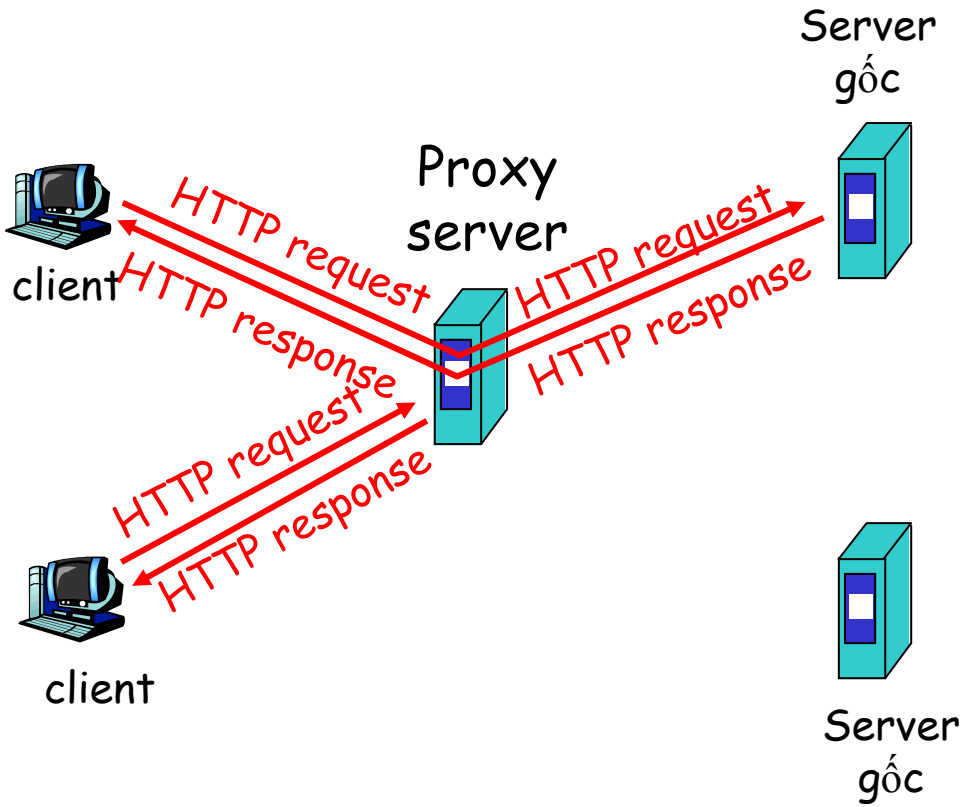
  HTTP/1.0 304 Not modified



Web Caches (Proxy Server)

Mục đích : Đáp ứng yêu cầu từ client mà không cần tới server thật sự chứa đối tượng (server gốc)

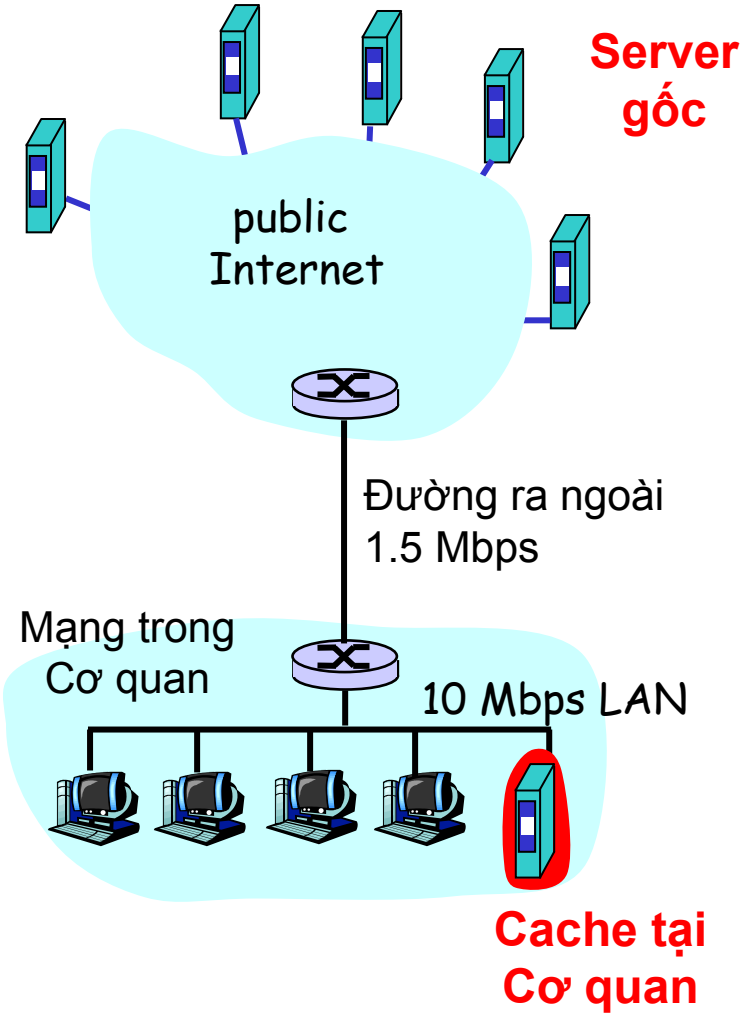
- Người dùng cấu hình browser truy nhập Web thông qua Web Cache
- Client gửi tất cả các yêu cầu HTTP request tới Web Cache
  - **Nếu có đối tượng**, Web Cache gửi đối tượng về cho client.
  - **Nếu không có**, Web Cache yêu cầu đối tượng từ server thực, sau đó gửi đối tượng cho client



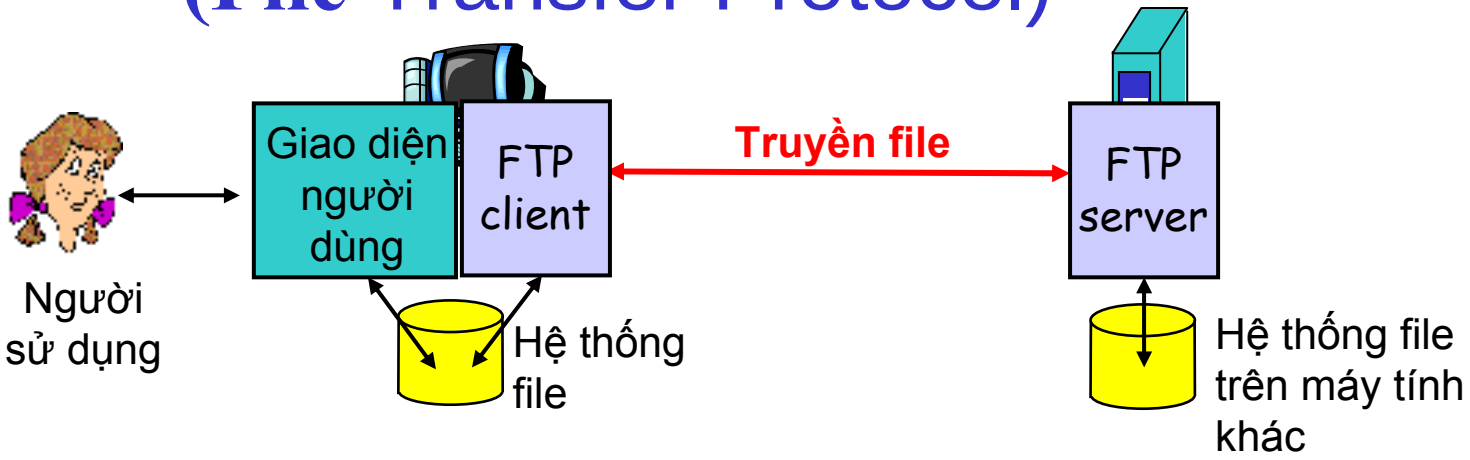
Tại sao sử dụng Web Caching ?

Giả định rằng : cache gần client hơn (có thể trong cùng mạng LAN)

- Giảm thời gian client phải đợi
- Giảm tải Mạng
  - Đường kết nối ra ngoài mạng nội bộ thường xảy ra hiện tượng "thắt nút cổ chai".



Giao thức Truyền file FTP (File Transfer Protocol)

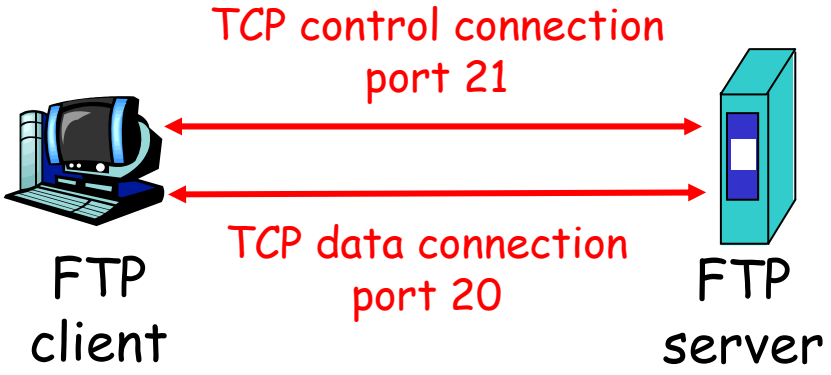


- Là giao thức trao đổi file với máy tính ở xa (remote host).
- Sử dụng mô hình client/server
  - *client:* khởi tạo kết nối.
  - *server:* máy tính ở xa (remote host)
- FTP được đặc tả trong RFC 959
- FTP server: port 21

Hai kênh kết nối FTP: Kiểm soát và Dữ liệu

- FTP client kết nối với FTP server qua cổng 21.
- Hai kết nối TCP đồng thời được tạo:
  - **Kiểm soát :** trao đổi lệnh, phản hồi giữa client và server.

    "out of band control"
  - **Dữ liệu :** kết nối tải file từ client đến server hay từ server về client.
- FTP server lưu lại trạng thái: thư mục hiện thời, lần truy nhập gần đây nhất



Các Lệnh và Trả lời thường gặp

Các lệnh thường gặp:

- Được mã hoá bằng mã ASCII
- **USER** ***username***
- **PASS** ***password***
- **LIST** trả về danh sách các file và thư mục trong thư mục hiện thời.
- **RETR filename** lấy file từ thư mục hiện thời.
- **STOR filename** Tải file vào thư mục hiện thời trên máy tính ở xa

Các mã trả về thường gặp

- Tương tự HTTP
- **331** chấp nhận username, yêu cầu password
- **125** kết nối dữ liệu được thiết lập, chuẩn bị truyền dữ liệu.
- **425** Không thể thiết lập kết nối dữ liệu
- **452** Lỗi ghi file.

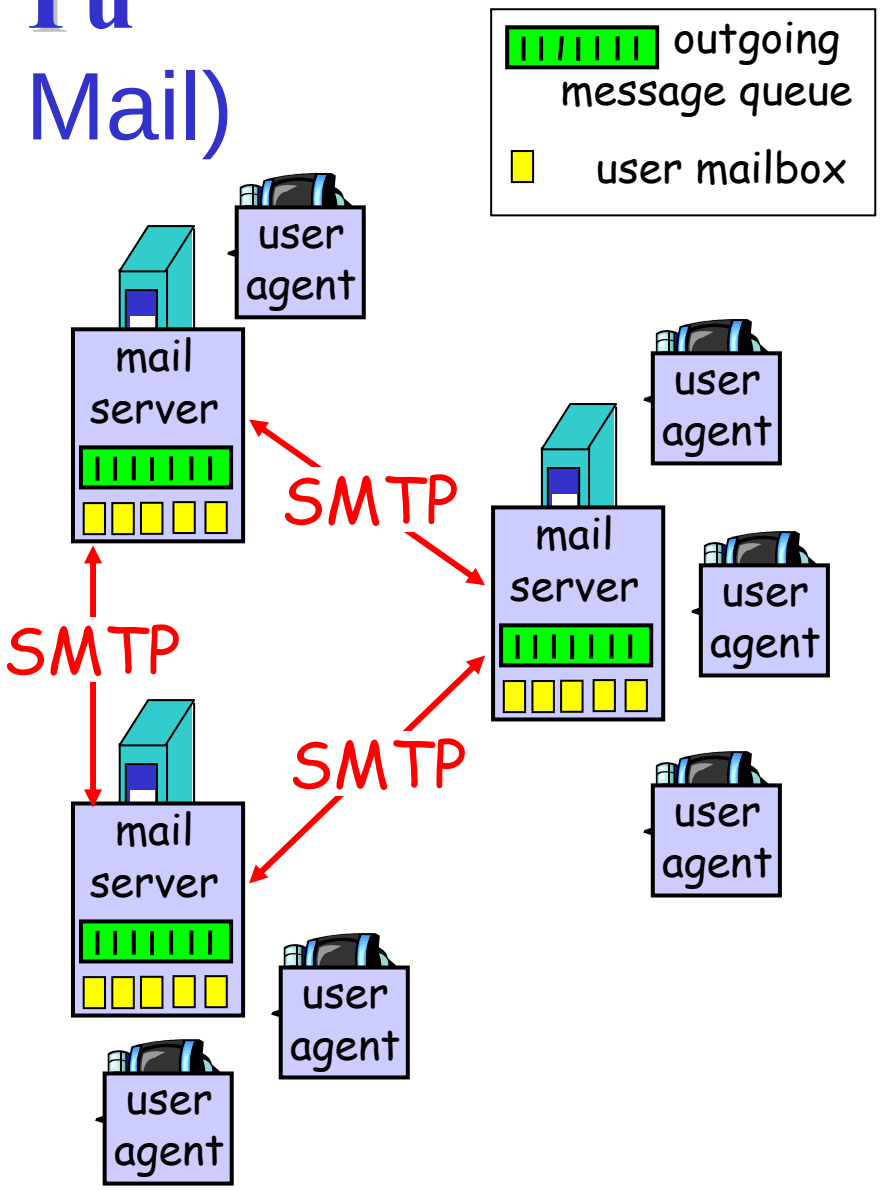
Thư Điện Tử (Electronic Mail)

Gồm 3 thành phần chính:

- User agent
- Mail server
- **S**imple **M**ail **T**ransfer **P**rotocol: **SMTP**

User Agent

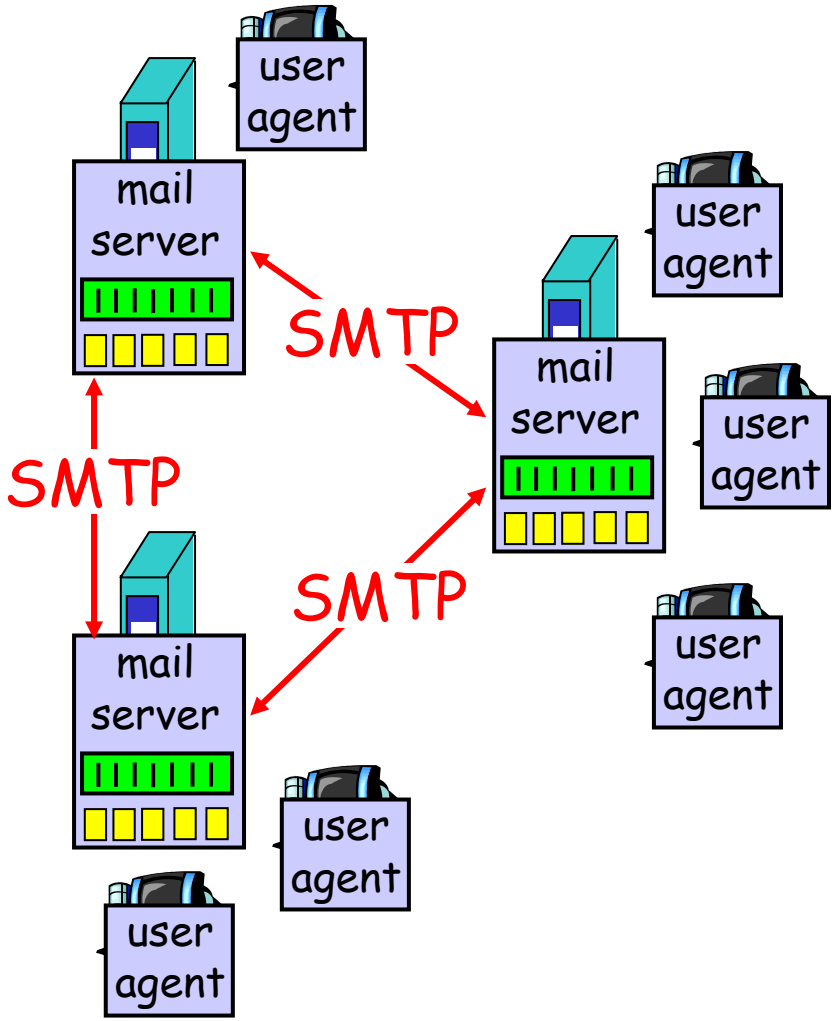
- Còn gọi là "mail reader"
- Soạn thảo và Đọc thư
- Ví dụ: Eudora, Outlook, elm, Netscape Messenger
- Các thông điệp outgoing, incoming được lưu trên server



Thư điện tử: Mail Server

Mail Server

- Hộp thư lưu lại các thư điện tử (có thể chưa được đọc) của người sử dụng.
- Hàng đợi chứa các thư sẽ được gửi đi.
- Giao thức SMTP là giao thức để các mail server trao đổi email.
  - Client dùng khi gửi thư.
  - Server dùng khi nhận thư.



Thư điện tử : Mail Server [RFC 821]

- Sử dụng dịch vụ TCP, truyền email tin cậy từ SMTP client tới SMTP server qua cổng 25
- Truyền trực tiếp : server gửi tới server nhận.
- Truyền qua ba giai đoạn :
  - "Bắt tay"
  - Truyền các thông điệp (Thư)
  - Đóng kết nối.
- Tương tác: Lệnh (client => server) và Trả lời (Server => Client)
  - **Lệnh** : mã bằng bảng mã ASCII.
  - **Trả lời** : mã trạng thái và có thể có thêm giải thích
- Các thông điệp phải được mã bằng bảng mã ASCII 7 bit

Một số tương tác SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Tự thử nghiệm tương tác bằng SMTP

- **telnet servername 25**
- Nhận được hồi đáp có mã 220 từ server.
- Đánh các lệnh **HELO, MAIL FROM, RCPT TO, DATA, QUIT**.
- Các lệnh trên cho phép bạn gửi thư mà không cần dùng đến email client (reader)

SMTP : Kết thúc

- Giao thức SMTP sử dụng kết nối bền vững.
- SMTP đòi hỏi thông điệp (header & body) phải được định dạng bằng mã ASCII 7bit.
- Những xâu kí tự đặc biệt (ví dụ CRLF.CRLF) không được phép ghi vào thông điệp (do đó chuỗi ký tự này phải được mã hóa)
- SMTP server sử dụng CRLF.CRLF để đánh dấu kết thúc thông điệp.

So sánh với HTTP:

- HTTP: giao thức kiểu "**kéo**" ("kéo" thông tin từ server về)
- Email :là giao thức kiểu "**đẩy**" ("đẩy" thông tin lên server)
- Cả hai đều tương tác gửi **lệnh** (mã ASCII)/ **trả lời** (mã trạng thái).
- HTTP : Mỗi đối tượng nằm trong một thông điệp riêng.
- SMTP: nhiều đối tượng nằm trong cùng một thông điệp

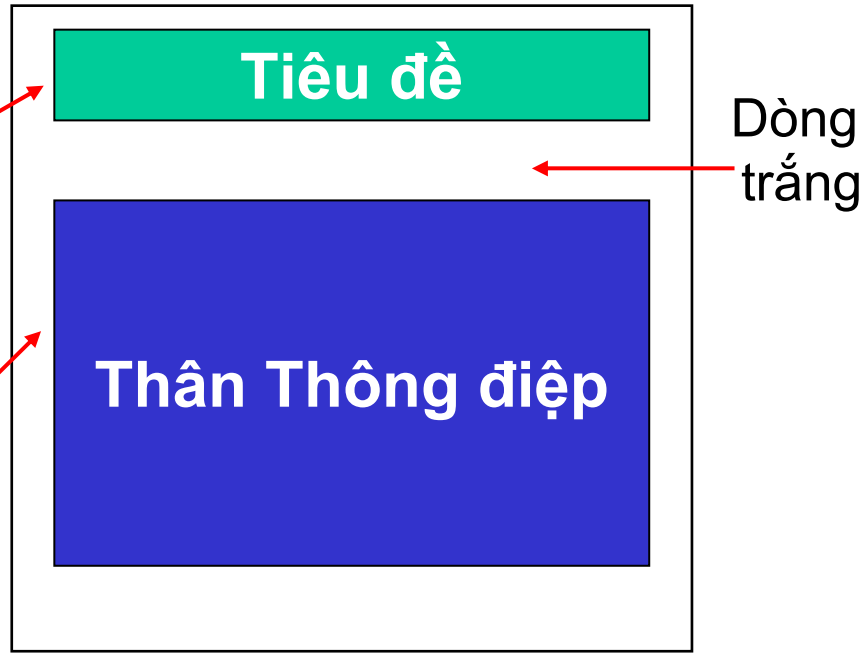
Định dạng Thông điệp eMail

SMTP : Giao thức để trao đổi email.

RFC 822 : chuẩn định dạng thông điệp email:

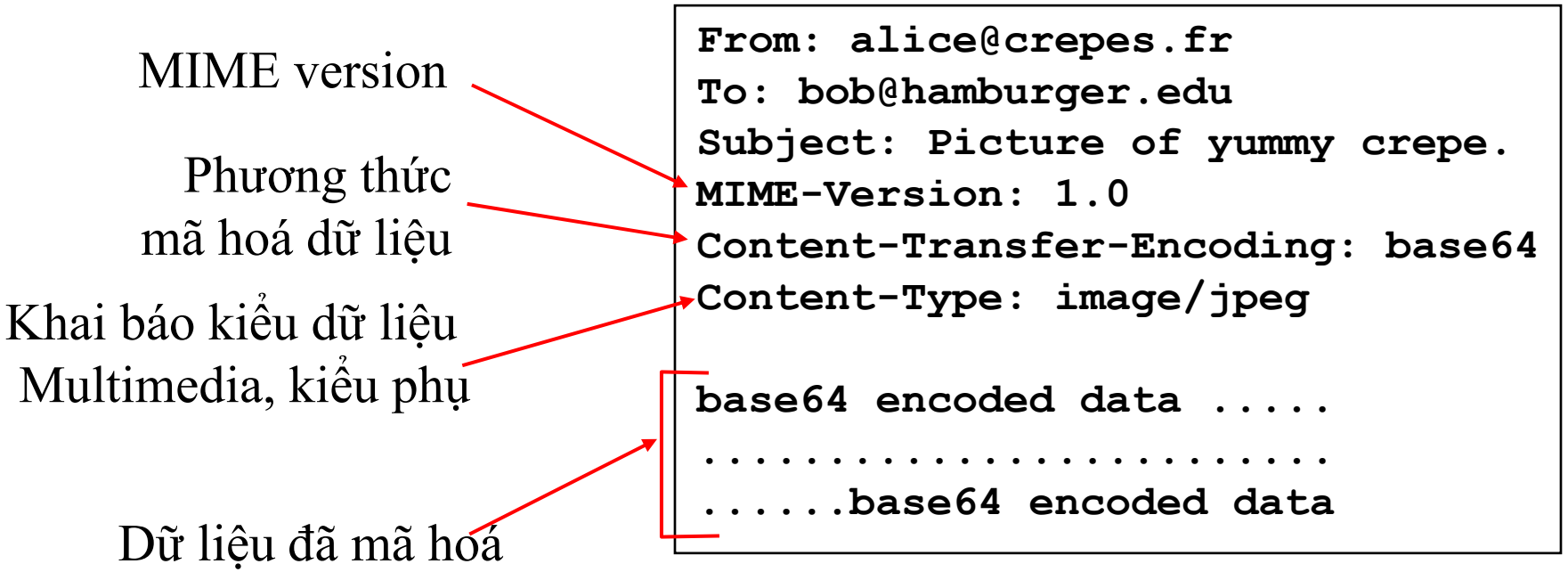
- **Tiêu đề Thông điệp :**
  - To:
  - From:
  - Subject:

  rất khác so với lệnh của SMTP !
- **Thân thông điệp**
  - Chỉ bao gồm các ký tự kiểu ASCII



Định dạng thông điệp mở rộng đa phương tiện

- **MIME** : Multimedia Mail Extension, RFC 2045, 2056
- Bổ sung thêm một dòng trong phần tiêu đề của thông điệp để mô tả kiểu MIME.



Kiểu MIME

`Content-Type: type/subtype; parameters`

Text

- Kiểu : **plain, html**

Image

- Kiểu : **jpeg, gif**

Audio

- Kiểu : **basic** (loại có quy luật hoá 8-bit), **32kadpcm** (mã hóa 32 kbps)

Video

- Kiểu : mpeg, quicktime

Application

- Các loại dữ liệu khác phải được xử lý bằng chương trình đọc tương ứng mới có thể đọc, xem được.
- Ví dụ các kiểu: msword, octet-stream

Kiểu chứa nhiều loại đối tượng

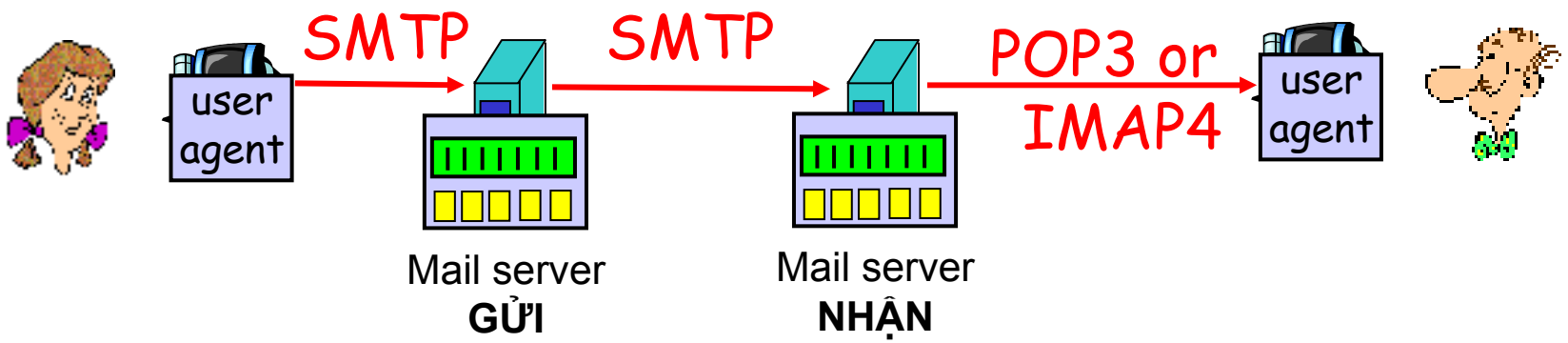
```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Các giao thức truy nhập Mail



- **SMTP:** Gửi thư tới Server chứa thư của người nhận
- Giao thức đọc thư : lấy thư từ server
  - **POP : Post Office Protocol [RFC 1939]**
    - Kiểm chứng (agent <-->server) và tải thư về
  - **IMAP: Internet Mail Access Protocol [RFC 1730]**
    - Phức tạp hơn do có nhiều đặc tính hơn.
    - Thao tác trên các thư lưu trên server.
  - **HTTP : Hotmail , Yahoo! Mail, Gmail,…**

Giao thức POP3

Giai đoạn kiểm chứng

- Các lệnh Client gửi:
  - **user:** username
  - **pass:** password
- Server trả lời
  - **+OK**
  - **-ERR**

Giai đoạn xử lý,cập nhật :

- **list:** in ra danh sách các thư(được đánh số ID).
- **retr:** lấy thư có ID là số nhập vào
- **dele:** xoá thư
- **quit**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```